OKRM DAG Scheduler

Math team

INTRODUCTION:

In real time application, race condition occurs when two or more queries trying to access the same column of the table. Multiple read queries can be executed concurrently because multiple reading of a column is independent of each other. If there is read-write, write-read, write-write queries accessing on same column then the problem occurs.

The query scheduling algorithm defined below uses arrays to resolve the conflicts by executing independent queries at same time and making dependent queries wait according to their time of arrival.

ALGORITHM

Step 1: Read column names and then apply hash functions (example: Universal hash function) to generate keys.

Syntax:

```
Read(Column Names)
{
Hash(Column Names)
} return (keys)
```

Step 2: The following function F takes incoming queries are returns the column names.

Syntax:

```
Function F (Queries)
{
return Column Names
}
```

Step 3: Create an array for each column and store incoming queries in respective arrays.

3.1 Creation of array

For example, A1[ ] for C1, A2[ ] for C2, A3[ ] for C3

Syntax: A1[ ], A2[ ], A3[ ]

3.2 Store the queries in array based on column name returned by function F.

Example : If Query 1 accessing column1, column 2, column 4.Query 2 accessing column1, column 2, column 3 and Query 3 accessing column 3.

Syntax:

Then store it as

A1[ ]={Q1,Q2,Q3} , A2[ ]={Q1,Q2}, A3[ ]={Q2,Q3}, A4[ ]={Q1}

Step 4: Resolving read-write , write-read, write-write conflicts

4.1 Swapping

- Query execution is done only at first index.
- Concurrently execution of queries in same column is restricted.
- During query execution, lock the corresponding columns
- Swap the queries in unlocked locks, to execute independent queries

Syntax :

Swap(A1[0],A1[j])

- A1[0]-fixed index position
- A1[j]-j varies till the independent query is found

example : swap(A1[0],A1[1])

4.2 Double swap

- If the queries is not executed after swapping , then again swap the queries to its original position
- Double swap is done to maintain the timing order of incoming queries

Syntax:

```
Swap(A1[0],A1[i])
If(independent query) :
    Execute the query
Else:
    Swap(A1[0],A1[i])
```

4.3 Swap count

- Maintain swap count for each column, to record number of swaps made
- Fix some limit to swap count

Syntax:

Fixed swap count=p; p is a positive integer

example:

Fixed swap count =5

- If swap count for any column exceeds than fixed swap count, then choose query which has minimum dependency and execute it first.
- Maintain the timestamp order while choosing minimum dependent query.

Step 5 : Repeat the Step4 until all the queries are executed.

Syntax:

```
While(arrays!=NULL)
{
Repeat STEP 4
}
```

Further Discussion

1. Arrays

- swapping of queries is done easily
- Parallel execution of queries can be done when using multiple arrays

-Constraint in array

Have to fix the size of an array, when the elements are to be entered into array.

In real time application it is difficult to predict the size of incoming queries.

-Alternative solution

Linked list- Dynamically the new nodes are created based on incoming queries

2.In other data structures

Tree- rotation of executable query to root node is difficult

Queue-swapping of queries cannot be done